# LELIE - An Intelligent Assistant for Improving Requirement Authoring

Patrick Saint Dizier, Juyeon Kang

**LELIE: An Intelligent Assistant for Improving Requirement Authoring**

## 1. LELIE project

LELIE[1] was funded by the French National Research Agency (ANR) from 2008 till 2013. It is still a research framework but it is now paired with R&D efforts in order to investigate its relevance and customization to the industrial world. The LELIE project is a research and R&D framework, based on natural language processing and artificial intelligence, the aim of which is to detect and analyze potential risks in technical documents, related to health and ecology, but also to a number of social and economic dimensions.

Risks emerge from poorly written texts, and from various forms of incoherence. For example, '*Progressively heat the probe X27'* relies too much on the operator's knowledge and practice: what temperature should be reached and in how much time? A wrong interpretation may lead to accidents of damages. Among technical documents, requirements are a central issue since they must comply with a high number of constraints of e.g. readability, lack of ambiguity and implicit data, feasibility, relevance, traceability, and overall cohesion and coherence.

The main aim of LELIE is, given a set of requirements, whatever their domain and type, to analyze their contents and to annotate them in an appropriate way wherever potential errors are identified. Errors range from poor writing quality to incoherence between requirements. Authors are then invited to revise these documents. This requires some domain knowledge for example: ontology, terminology, and lexical. Requirements are a textual genre dedicated to action: little space should be left for ambiguities and for personal interpretation.

LELIE is based on three levels of analysis:
• The detection of inappropriate ways of writing requirements: lexical inadequacies (e.g. fuzzy terms, misuses of business terms), complex constructions (e.g. connectors, conditionals, stacks of nouns), complex references, inappropriate granularity level, etc.
• The detection of various types of incoherence: incoherence among sets of requirements, incoherence with respect to domain knowledge or practices (e.g. use of unusual instruments, equipment, product, or unusual values such as a too low or too high temperature) as e.g. specified in other technical documents,
• Confrontation of domain safety requirements with procedures to check if the required safety constraints are met. For example, when manipulating an acid, check that the operator has protection gloves and glasses, as required.

The LELIE project addresses a large number of problems of controlled natural language. We concentrate in this document on the first topic: the detection of inappropriate ways of authoring requirements, which has now reached a good level of maturity. A prototype has been developed for this first topic for French and English. A kernel of this prototype, without any fancy interface facility, is available for testing at: http://www.irit.fr/~Patrick.Saint-Dizier/. The two other topics given have reached a lower level of maturity: they are extremely complex in general. Investigations are made on a case-based approach.

[1] The name LELIE is not an acronym, it is a character from Molière who makes a lot of errors in his everyday life, hence the name for our project.

The approach in LELIE is not to guide requirement authors to write on the basis of predefined templates, also called boilerplates, which are not very often strictly followed, but to let authors express themselves freely and then to make, upon demand, a posteriori controls.

LELIE develops a hybrid approach that is cooperative with the requirement author based on:
**(1)** The use of error templates to detect errors typical of requirements, which may not be errors in ordinary language. These errors are defined on the basis of (1) the Controlled natural Language (CNL) principles (Kuhn 2014) paired with (2) various authoring guidelines produced by companies, which are in general relatively coherent with CNL principles and complement them. For each potential error, the system produces alerts with some explanation. CNL principles are composed of a set of constraints on the structure of sentences, paragraphs, titles and on the type of vocabulary which can be used.
**(2)** The association of this first level, based on fixed templates, with an error correction memory, adds flexibility and context to templates in order to limit noise from the first stage (e.g. a fuzzy term is fuzzy only in certain contexts). The other goal of this second level is, via the observation of how authors make corrections or decide that an alert is irrelevant, to induce types of corrections in order, after validation, to propose them in a later stage. This greatly reduces writers' workloads and also establishes correction norms over a team of technical writers, resulting in documents which are much more homogeneous.

Tools controlling the authoring quality of requirements have been developed in the past with the use of templates or boilerplates meant to guide the technical writer (Arora et al. 2013). This is most notably the case for the well-known RAT-RQA system (the Reusecompany) and of the RUBRIC system developed at the University of Luxemburg. Let us also cite two major CNL-based university prototypes which are of much interest for requirement authoring: ACE (Fuchs et al. 2008, 2012), which stands for Attempto Controlled English. This system makes an in-depth language semantic analysis. It was initially designed to control software specifications, and has been used more recently in the semantic web. PENG (Processable English (White et al. 2009)) is a computer-processable controlled natural language system designed for writing unambiguous and precise specifications. These systems make heavy use of syntactic analysis, which is rather costly. LELIE is based on shallow parsing techniques and semantic analysis, which makes it more relevant for requirements where the language is complex and sometimes ill-formed. A synthesis of CNL based systems is developed in (Kuhn 2013, 2014).

## 2. LELIE systems

### 2.1 Error detection tool

Let us now concentrate on LELIE as an intelligent assistant tool for requirement authoring. LELIE is a system based on rules that detect errors of different levels: syntactic, lexical, semantic, discourse. From the analysis carried out by LELIE, it becomes easier to measure the quality of a specification, composed of requirements, in terms of its testability, ambiguity, singularity, consistency, completeness, redundancy and traceability. The error correction rules have been developed and validated in four steps:

1) First, authoring rules proper to requirements have been collected and summarized from the IEEE standards, the specific recommendations for requirements authoring like the guide proposed by INCOSE and the principles of controlled languages like the Simplified Technical English (STE) defined by the ASD. In these sources, we observed

that the authoring constraints specify the syntax, the semantics together with the style and the lexicon that the authors have to observe. The generic rules of LELIE have been identified from this analysis and constitute the basis of our model.

2) Then, we observed local practice in various companies which often have their own set of authoring recommendations. This was realized essentially via the observation of technical writers at work and via discussions on their authoring strategies (Barcellini et al. 2012). As a result, the generic rules established at step 1) have been complemented by these more local rules and the potential inconsistencies that may arise.

3) Next, we analyzed sets of requirements written by various authors, mainly in the electricity, telecommunication, aeronautics and automobile domains, in order to analyze if and how these rules were used in concrete situations. This analysis was carried out on the usual language levels: lexical, syntactic, semantic, and discourse. A requirement normally consists of a condition, a subject, an action, an object and constraints. The discourse analysis provides the capability to characterize the semantic relations that hold between the components of a requirement which are not just subject or objects.

4) Finally, feedback from users (Schriver 1989) were very important to validate, improve and enrich the rules and the lexicon, such as fuzzy terms, buzz words to avoid, and, obviously, business terms appropriate to a specific domain.

Table 1 illustrates the main types of errors found in a set of procedures, requirements, and safety specifications. These documents come from three companies from different domains of industry: energy, aeronautics and car manufacturing. 60 pages were considered in this analysis. The error frequency is indicative: it may vary largely depending on the activity and the type of document.

| **Types** | **Identified problems/ Error frequency** | **Examples** |
|---|---|---|
| Fuzzy terms | ambiguity, testability 25% | *wherever possible, suitably, adequately* |
| Complex or ambiguous coordination | Singularity 8% | *X shall* ACTION1 *and* ACTION2 *or* ACTION3 |
| multiple negation makers or double negation | Readability 8% | *It shall not be possible to do not…* |
| Multiple actions in a requirement | validity, testability, traceability 6% | *X shall* ACTION1 *and* ACTION2 / *X shall* ACTION1 *and Y shall* ACTION2 |
| Complex relatives | Readability 7% | *that x…which*…y…. |
| Complex discourse structures | readability, ambiguity 15% | [SUBJET],-[CONDITION],-[ACTION]-[OBJECT] |
| Pronouns with uncertain reference | Ambiguity 8% | *their, them, these, it…* |
| Incorrect references to | Not feasible | *below, above, see…* |

| | | |
|---|---|---|
| other chapters | 3% | |
| Heterogeneous enumeration | Waste of time to understand, ambiguity 20% | *a. system interaction*<br>*b. user interface*<br>*c. update is not applicable*<br>➔ non homogeneous with a and b |

Table 1. Rule description

The texts of the company S3 have been reviewed by experts of technical document production before our analysis, however there remain several errors. We observe that the distribution of the errors depends in particular on the complexity of texts: those of S2 are clearly more complex than those of S1. Finally, we note that there are on average 15 errors by page, i.e. approximately an alert every 2 or 3 lines, not counting errors related to the business rules. This is obviously very large and motivates the use of LELIE.

### 2.2 The error correction memory

The alerts produced by the LELIE system have been found useful by most requirement writers that tested the system. However, they feel that:
-false positives, about 40% of the alerts, must be filtered out. This is essentially due to the lack of context sensitivity of error detection rules, e.g. *progressively* in *shall progressively close the pipe* is judged not to be fuzzy because the action is short, whereas it is fuzzy in *shall progressively decrease the air speed*.
-errors which are not detected, about 8% to 25%, should be reduced as much as possible to guarantee a good performance level. Non-detection originates e.g. from incomplete lexical resources in the system.
- Error severity levels must be finely tuned so that requirement writers can organize their revisions, starting e.g. by the most severe errors. Indeed, an error detection system must be very flexible with respect to the writer's practices.
- Help must be provided in the form of e.g. correction recommendations, whenever possible.
- Corrections should be memorized so that they can benefit others and, in the long term, allow homogeneous corrections over a whole team of authors. These could also be re-used as a tutoring system for novices.

In the LELIE project, we develop and test several facets of an error correction memory system that would, after a period of observation of requirement writers making corrections from the LELIE alerts, add flexibility and context sensitivity in error detection and correction. General principles of language processing via a contextual memory are developed in (Daelemans 2005).
This memory system is based on the following operations:
- Memorize errors which are not or almost never corrected so that they are no longer displayed as errors in the future: these are called false positives,
- Memorize corrections realized by writers, with their context,
- Automatically induce typical corrections, proper to requirement styles,
- Organize a correction validation process to produce correction recommendations. This is managed by an administrator or via mediation in a group of writers.

The error correction memory is based on a two level organization:

(1) The development of relatively ***generic correction patterns***, which correspond to a common correction practice for most types of requirements. These are stable over a domain, a company or a type of requirement. These patterns are induced from the general behavior of requirement writers when they make corrections. They often contain underspecified fields.
(2) The development of accurate ***contextual correction recommendations***, based on previously memorized and analyzed corrections. Recommendations are induced from a small set of closely related terms and situations in context. These are paired with the generic correction patterns: they suggest values for the underspecified fields.

Roughly, after induction (step (1) above), an error correction rule has the following form:
**[error pattern] → [correction pattern] – Context**.
The "error pattern" describes an incorrect structure, the "correction pattern" is the correction that should preferably be applied, while "Context" refers to the conceptual environment of the correction pattern. In LELIE it is realized by memorizing the four closest words (adjectives, nouns, verbs) occurring before or after the error. The context allows the specification of precise recommendations.

For example, a correction rule used for fuzzy manner adverbs is.:
**[*progressively* VP(durative)]} → [*progressively* VP(durative) in X(time)] – Context.**
where X(time) is a variable of type time. VP(durative) indicates an action that takes some time to be realized.
e.g. *progressively heat the probe X37* → *progressively heat the probe X37 in 10 minutes*.
In this example, Context = (*Probe X37 heat*), VP = *heat* and X= *10 minutes*. X is suggested by a correction recommendation in relation with the context (*heating the X37 probe*), the adverb is kept in order to keep the manner facet which is not fuzzy, since it is the temporal dimension that is fuzzy. Note that '*heat'* is here underspecified: the temperature to reach is not given. This is another type of error detected by LELIE, but not developed in this text.

We noted that correction divergences between technical writers often arise; therefore, a strict automatic learning process is not totally accurate and achievable. In LELIE, the approach is to propose to a team of technical writers several possible corrections via simple generalizations on coherent subsets of corrections and to let them decide on the best solution, via discussion, mediation, or via a decision made by an administrator.

Let us now concentrate on a few typical cases related to fuzzy terms and negation, which are frequent errors in requirement authoring. There are several categories of fuzzy lexical items which involve different correction strategies. They include a number of adverbs (manner, temporal, location, and modal adverbs), adjectives (*adapted, appropriate*), determiners (*some, a few*), prepositions (*near, around*), a few verbs *(minimize, increase*) and nouns. These categories are not homogeneous in terms of fuzziness, e.g. fuzzy determiners and fuzzy prepositions are always fuzzy whereas, for example, fuzzy adverbs may be fuzzy only in certain contexts. The degree of fuzziness is also quite different from one term to another in a category.

On a small experiment with two technical writers from one of our users, considering 120 alerts concerning fuzzy lexical items in different contexts, 36 have been judged not to be errors (rate: 30%). Among the other 84 errors, only 62 have been corrected. The remaining 22

have been judged problematic and very difficult to correct. Correcting fuzzy lexical items indeed often requires domain expertise.

To conclude this section, let us give a few typical error correction patterns that have been induced, a number of them deal with various forms of implicit quantification:

| Error type | Error pattern | Correction pattern | Example |
|---|---|---|---|
| **Fuzzy determiner** | **[*a few* Noun]** | **[*less than* X Noun]**<br>*Adds an upper boundary X | *A few minutes --> Less than 5 minutes* |
| | **[*most* Noun]** | **[*more than* X Noun]**<br>*Adds a lower boundary X | *Most pipes shall ...*<br>*→More than 8 pipes shall...* |
| **Temporal, iterative adverbs** | **[VP(action) Adverb(iterative)]**<br>*VP(action): action verb | **[VP(action) *every* X(time)]** | *The steam pressure shall be controlled regularly*<br>*→The steam pressure shall be controlled every 10 minutes.* |
| **Fuzzy prepositions** | **[*near* Noun(location)]** | **[*less than* X(distance) *from* Noun(location)]**<br>*X(distance) depends on Context | *Near the gate →*<br>*Less than 100 m from the gate* |
| **Negation on usages** | **[*(do) not* Verb(*use*) NP]**<br>*NP: any noun<br>*Verb(*use*) any verb such as 'use' | **[Verb(use) hyperonym(NP) other than NP]**<br>*Hyperonym(NP) denotes a more generic term than the NP, given in a domain terminology | *shall not use hydrogen*<br>*→shall use a gas other than hydrogen* |
| **Reverse synchronization** | **[*do not/never* VP *before* VP']**<br>*VP and VP' denote two actions | **[VP only after VP']** or **[VP'. Then VP]**<br>*Actions are reversed in the correction, some persuasion effects may be lost. | *never unplug before the machine has been stopped.*<br>*→stop the machine. then unplug it* |

Table 2. Error patterns and related Correction patterns

## 3. LELIE system architecture and the prototype

The LELIE prototype is based on the following components:

- An engine, TextCoop, that manages the different parsing and the enforcement of linguistic constraints. The engine is domain independent. TextCoop is an engine

developed at IRIT[2] for text and discourse processing in general (Saint-Dizier 2014)..

- A set of rules that handle the different types of alerts described above. Rules are a priori domain independent, however some may be tuned or skipped depending on a user's needs or company guidelines
- A lexicon in the language considered. Functionally, the lexicon is decomposed into two units: (1) the main one corresponds to ordinary language - it is generic and is used in any application (possibly with some minor adjunctions) - and (2) a secondary one that contains all the terms specific to a domain; in particular the ontology of business terms is stored in this latter lexicon.
- A set of utilities, I/O facilities, etc., and
- A set of parameters to tune the system, e.g. choosing which rules to apply.

The kernel of the system is the set of rules that consult lexical entries. The lexical entries of the secondary lexicon must be defined for each domain (e.g. aeronautics, energy, chemistry) and possibly adapted or tuned for each application. This can be done manually, by lexicographers, or via the support of a lexical acquisition platform. The kernel is available for testing.
The LELIE architecture is summarized in Figure 1:

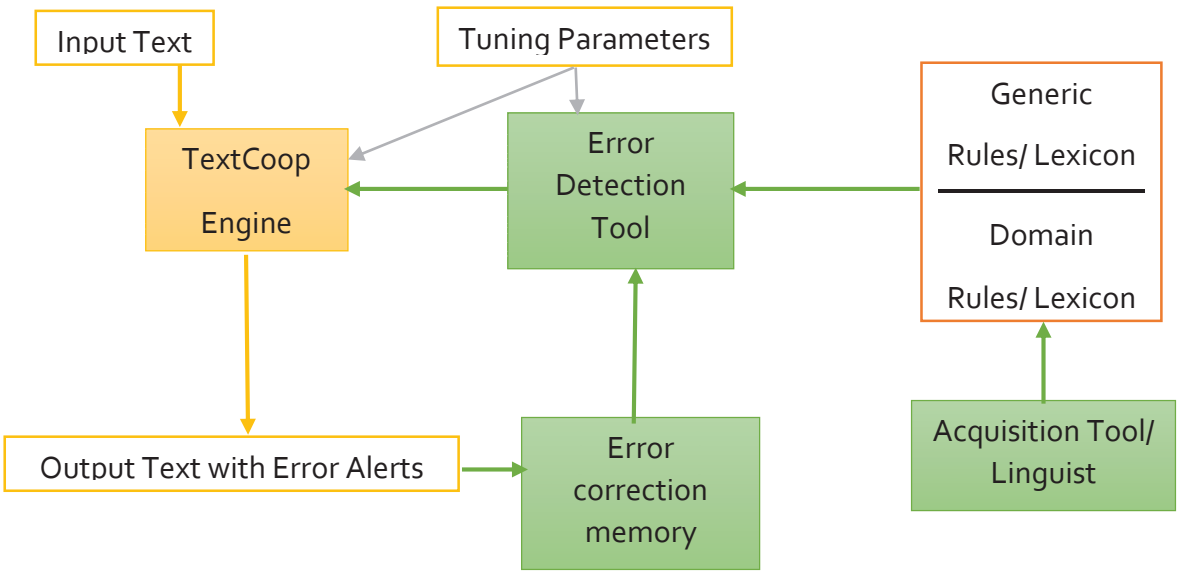


Figure 1. LELIE Architecture

## 4. Perspectives

LELIE is a project which aims to investigate the different tools that are needed for technical writers, and in particular requirement writers, to improve the quality of their texts. We have presented in the previous section the first step: improving the authoring quality of texts, via alerts and correction patterns. This is the first step in such a project. It is important to note

[2] Institut de Recherche en Informatique de Toulouse : www.irit.fr

that, although there are guidelines for writing requirements, large differences in style and form have been observed between authors and companies.

Once requirements are relatively well-written, additional quality controls can be carried out. Let us review here those which seem to be the most crucial from the errors found in large collections of requirements. Most of them are complex and cover several situations. Therefore, we feel a case-based approach is appropriate to analyze and develop them gradually in a sound way. These controls are, in particular:

- ***The analysis of the cohesion of a set of requirements***: lexical, grammatical, and style cohesion is a plus since it makes long lists of requirements easier to read. Lexical cohesion requires, e.g. a strict control of the terms used: a concept is always referred to by the same word or expression.
- ***The detection of forms of clumsiness***, in particular when authors do not write in their mother tongue, for example, French authors writing in English produce typical clumsy forms that need to be revised, although they are not errors as such.
- ***The analysis of forms of redundancy*** over large sets of requirements: redundancy can be partial or complete.
- ***The detection of forms of partial incoherence*** over sets of requirements. This latter task is very challenging and may require some form of domain knowledge.
- Finally, concerning security requirements, which are a specific class of requirements, a useful operation is to check that these requirements are met in related procedures. For example, the precautions to take when manipulating an acid are specified at the right place in any procedure that requires the manipulation of such an acid.

## 5. References


Alred, G.J., Charles T.B., and Walter E.O., *Handbook of Technical Writing*. St Martin's Press, New York, 2012.

Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F., Gnaga, R., Automatic Checking of Conformance to Requirement Boilerplates via Text Chunking: An Industrial Case Study, 7th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2013). Baltimore, MD, USA 2013.

Barcellini F, Albert, C., Saint-Dizier, P., 'Risk Analysis and Prevention: LELIE, a Tool dedicated to Procedure and Requirement Authoring', Language Resources and Evaluation Conference (LREC), Istanbul, 2012.

Daelemans, W., van Der Bosch, A., Memory-Based Language Processing, Cambridge, 2005.

Fuchs, N.E., 'First-Order Reasoning for *Attempto* Controlled English', In Proceedings of the Second International Workshop on Controlled Natural Language (CNL 2010), Springer, 2012.

O Grady, J. *System Requirements Analysis*, Academic Press, USA, 2006.

Kuhn, T., 'A Principled Approach to Grammars for Controlled Natural Languages and Predictive Editors'. ,Journal of Logic, Language and Information, 22(1), 2013

Kuhn, T., 'A Survey and Classification of Controlled Natural Languages'. Computational Linguistics, 40(1), 2014.

Saint-Dizier P, *Challenges of Discourse Processing: the case of technical documents*. Cambridge Scholars, UK, 2014.

Schriver, K. A., 'Evaluating text quality: The continuum from text-focused to reader- focused methods' IEEE Transactions on Professional Communication, 32, 238-255, 1989.

Unwalla, M., AECMA Simplified English, 2004. http://www.techscribe.co.uk/ta/aecma- simplified-english.pdf.

White, C., Schwitter, R.: An Update on PENG Light. In: Pizzato, L., Schwitter, R. (eds.) Proceedings of ALTA 2009, Sydney, Australia, pp. 80-88, 2009.

Wyner, A., et ali. On Controlled Natural Languages: Properties and Prospects, University of Aberdeen report, 2010.